

### REMARKS

The applicant's remarks, below, are preceded by quotations of related comments of the examiner, in small, bold-face, type.

**Claim 1 is rejected under 35 U.S.C. § 103 as being unpatentable over Antes, Gary M., "Let your 'knowbots' do the walking," Computerworld, May 13, 1991, pp(2), in view of Steinberg, Don, "Demon knowbots (intelligent software robots)," PC-Computing, v3, nl, pp(4), Jan, 1990.**

**With respect to claim 1, The Examiner notes that "administrative Knowbots" that "police the system, keeping unauthorized users out" necessarily must intercede between system access requests (e.g., from other "Knowbots") and the underlying local system service facilities. Accordingly, the rejection of claim 1, as set forth in the last office action, is maintained. Additional new grounds of rejection are also set forth for claim 1, as detailed below.**

**Antes discloses the invention substantially as claimed:**

**Antes teaches a method for use in a distributed system for processing a mobile program that has the ability to move from node to node in the distributed system [e.g., page 1, line 24].**

**Antes teaches an operating environment in each of the nodes that provides service facilities (e.g., databases) useful to the mobile program [e.g., page 1, line 30].**

**However, Antes does not explicitly disclose the following additional limitations:**

**Steinberg teaches an operating environment running a supervisor process [e.g., administrative knowbots, page 3, line 3] that allows the mobile program indirect access to make use of the service facilities [page 3].**

**It would have been obvious to one of ordinary skill in the art at the time the invention was made to improve upon the system taught by Antes by implementing the improvements detailed above because it would provide Antes's system with the enhanced capability of keeping unauthorized users out [e.g., page 3, line 4].**

The applicant respectfully disagrees. Without conceding any of the examiner's points, the applicant notes that, in claim 1, the supervisor process "allows the mobile program indirect access to make use of the service facilities." (emphasis added) Conversely what is said in the Steinberg article is that the administrative Knowbots police the system by "keeping unauthorized users out." It would not have been obvious to a person of ordinary skill in the field to run a supervisor process that allows indirect access by a mobile program to service facilities (as in claim 1) when the advice provided in the Steinberg article was to keep unauthorized users out, in

other words, excluded directly. The claim recites a method that provides access in a certain way. Steinberg describes a system that prevents access. If anything, Steinberg's statement teaches away from the invention recited in claim 1.

**Claims 1, 24 & 25 are rejected under 35 U.S.C. § 102(a) as being anticipated by Vanderburg, Glenn L., et al., "Tricks of the Java Programming Gurus," Sams.net Publishing, see Chapter 33, entire chapter, pp(14), 1996.**

**As per claim 1:**

**Vanderburg teaches a method for use in a distributed system for processing a mobile program that has the ability to move from node to node in the distributed system [e.g., see "Java" discussion beginning page 1]. The Examiner notes that mobile Java programs (i.e., applets) anticipate the claimed limitations.**

**Vanderburg teaches an operating environment in each of the nodes that provides service facilities useful to the mobile program [e.g., a Java Virtual Machine; see also Java security and native method discussion beginning page 1],**

**Vanderburg teaches an operating environment running a supervisor process [i.e., the Java Virtual Machine and associated security discussion, beginning page 1 ] that allows the mobile program indirect access to make use of the service facilities [e.g., see "ExtensibleSecurityManager.java" code listing and associated native method discussion beginning page 5 ]. Java byte codes "indirectly interact" via the operating environment, as they are interpreted by the Java virtual machine at run time.**

The applicant has amended this application to claim priority from a co-pending United States patent application that was filed on May 30, 1995. Portions of the parent application that describe examples of what is recited in claim 1 have been copied into this application and other portions of the parent application are incorporated by reference. Thus, with respect to at least claims 1, 24, and 25, the applicant's effective filing date precedes the publication date of the Vandenburg article, and the rejection is moot.

**As per claim 24:**

**Vanderburg teaches a method for controlling interaction between a mobile program and an application running in an operating environment provided at a node of a distributed system comprising:**

**defining a trusted portion of the operating environment which provides trusted services to the mobile program [e.g., see discussions of "Security in Native Method Libraries," page 1, and the "Extensible Security Manager," beginning page 4 - see code listing page 5-10],**

requiring portions of the application running in the operating environment to be registered as trusted, [e.g., see "registered" code as marked (underlined by Examiner) in the code listing pages 7-10], and

permitting indirect interaction via the operating environment between the mobile program and the application running in the operating environment only if the portions of the application required to be registered have been registered [e.g., see "registered" code as marked (underlined by Examiner) in the code listing pages 7-10; e.g., see "Registering Specialized Security Managers" discussion beginning page 12]. The Examiner notes that the code listing is dated on pages 5 and 6 as Feb. 31, 1996. Java byte codes "indirectly interact" via the operating environment, as they are interpreted by the Java virtual machine at run time.

Please see the discussion above, which applies to claim 24.

As per claim 25:

This claim is rejected for the same reasons detailed above in the rejection of claim 24, and also for the following additional reasons:

Vanderburg teaches a method for enabling a mobile program to carry out defined functions including otherwise unsafe functions, though the use of extensions comprising:

coding safe extensions to an operating environment and to an interpretive language under which the mobile program runs, [e.g., see "ExtensibleSecurityManager.java" code listing beginning page 5, entire code listing and supporting discussion], and

permitting the mobile program to carry out the defined functions by making use of the extensions [e.g., see "registered" code as marked (underlined by Examiner) in the code listing pages 7-10; e.g., see "Registering Specialized Security Managers" discussion beginning page 12].

Please see the discussion above, which applies to claim 25.

In light of Applicant's arguments of record, the Examiner has reconsidered and withdrawn the remaining rejections. Dependent claim 2 appears to be allowable if rewritten to include all of the limitations of base claim 1, as the prior art does not appear to teach nor suggest the same or equivalent structure and function of the claimed "bastion object." This claimed software component is accorded the status of a "coined term" by the Examiner. Likewise, the claimed "connector objects" of claim 9 are examined in the context of a lexicon created by Applicant.

Accordingly, the scope of the aforementioned claimed coined terms is interpreted by the Examiner as being limited by the corresponding structure and function disclosed within the instant specification.

The applicant appreciates the indication that these claims would be allowable.

Without conceding any of the examiner's points, the applicant has amended claims 9 through 12 to remove the word "connector" and to make clear that the claim encompasses not only the connector as described in the specification, but any mechanism and objects that are within the scope of the claim language, including equivalents. The applicant contends that the claims are patentable as amended.

**Claims 5-17 appear to be allowable, subject to the results of a final search.**

The applicant acknowledges that claims 5-17 appear to be allowable.

**Claims 18 & 19 are rejected under 35 U.S.C. § 103 as being unpatentable over Antes, Gary M., "Let your 'knowbots' do the walking," Computerworld, May 13, 1991, pp(2), in view of Steinberg, Don, "Demon knowbots (intelligent software robots)," PC-Computing, v3, nl, pp(4), Jan, 1990, and further in view of Orfali et al., "Client/Server Programming with CORBA Objects," OS/2 Magazine, Sept. 1994, pp(8).**

**As per independent claim 18:**

**Antes, as modified by Steinberg, teaches the invention substantially as claimed.**

**Antes, as modified by Steinberg, teaches a method for aiding communication with a mobile program executing in operating environments provided at nodes of a distributed system (as discussed above in the rejection of claim 1).**

**However, Antes & Steinberg do not explicitly disclose the following additional limitations:**

**Orfali teaches maintaining a name space that uniquely identifies types of information to be interchanged as part of the communication [e.g., page 3, #7, i.e., "Register the run-time objects with the implementation repository" - see the disclosed "object reference" ], and using a name within the name space to identify a type of information to be interchanged [e.g., page 3, #7, i.e., "Register the run-time objects with the implementation repository"].**

**It would have been obvious to one of ordinary skill in the art at the time the invention was made to improve upon the combined system taught by Antes & Steinberg by implementing the improvements detailed above because it would provide their system with the enhanced capability of knowing which object classes are supported on a particular server [Orfali, page 3, discussion #7].**

**As per claim 19:**

**Antes, as modified by Steinberg and Orfali, teaches the mobile program registers an interface which includes the name of a type of information that is to be interchanged [e.g., Orfali, page 3, #7, i.e., "Register the run-time objects with the implementation repository"].**

The applicant discussed Antes and Steinberg in its prior response. With respect to Orfali, the applicant's claim 18 requires "using a name space that uniquely identifies types of

Applicant : Robert E. Kahn et al.  
Serial No. : 08/720,092  
Filed : September 27, 1996  
Page : 16

Attorney's Docket No.: 06154-008001

information to be interchanged." The CORBA approach described by Orfali records the type of an object being substantiated on a server for the purpose of matching client software with service interfaces which are generally used locally and which not interchanged or mobile.

The dependent claims are patentable for the same reasons as the independent claims on which they depend.

Applicant asks that all claims be allowed. Please apply any excess charges or credits to Deposit Account No. 06-1050.

Respectfully submitted,

Date: \_\_\_\_\_

*August 18, 92*



\_\_\_\_\_  
David L. Feigenbaum  
Reg. No. 30,378

Fish & Richardson P.C.  
225 Franklin Street  
Boston, Massachusetts 02110-2804  
Telephone: (617) 542-5070  
Facsimile: (617) 542-8906

amendment1.doc

Version with markings to show changes made

1. (previously amended) A method for use in a distributed system for processing a mobile program that has the ability to move from node to node in the distributed system comprising

in an operating environment in each of the nodes, providing service facilities useful to the mobile program, and

in the operating environment, in each of the nodes, running a supervisor process that allows the mobile program indirect access to make use of the service facilities.

2. (previously amended) The method of claim 1 further comprising creating a bastion object in an unrestricted environment to protect the unrestricted environment and passing it into a restricted environment within which the mobile program is running.

3. (previously amended) The method of claim 2 in which the bastion object provides an interface for the mobile program to access the service facilities in a safe manner and which is substantially the same interface as the interface that the service facilities provide in the unrestricted environment.

4. (previously amended) The method of claim 2 in which the bastion object performs type checking on all method calls made by a mobile program to a service facility.

5. (allowable) A method for use in a distributed system for processing a mobile program that executes in one node of the distributed system, may be interrupted at almost any point in its execution, and may be moved to another node of the distributed system for further execution, comprising

in the one node, capturing a current state of the mobile program execution,  
delivering the captured state and program code of the mobile program to the other node,  
and

continuing execution at the other node from the point of interruption based on the captured state and the program code.

6. (allowable) The method of claim 5 further comprising  
also delivering with the captured state and the program code a transported file system or other information created during execution of the mobile program.

7. (allowable) The method of claim 6 in which the information in the transported file system or other information is accessible without executing the mobile program.

8. (allowable) The method of claim 5 in which the step of capturing comprises using an encoding scheme of a language interpreter.

9. (twice amended) A method for enabling communication with a mobile program running in a distributed system, a mobile program service station, an extension, or another application, comprising

providing a [connector] mechanism which permits each of mobile program, the mobile program service station, the extension, and the other application to identify services that it provides, and permits each of them to find services that it needs, and

enabling the mobile program to communicate with mobile program service stations via [connector] objects associated with the [connector] mechanism.

10. (twice amended) The method of claim 9 in which each of the [connector] objects is provided by a supervisor process running in the distributed system and prevents uncontrolled access to a needed service.

11. (amended) The method of claim 9 in which the [connector] mechanism includes a [connector] broker and a [connector] manager.

12. (amended) The method of claim 9 in which the [connector] objects are data typed.

13. (previously amended) A method for enabling negotiation between two unrelated mobile programs, mobile service stations, extensions, or other applications, in a distributed system, comprising

in an operating environment in a node of the distributed system, receiving information from one of the two mobile programs, mobile program service stations, extensions, or other applications, concerning a transaction offered to other mobile programs, mobile program service stations, extensions, or other applications,

in the operating environment in the node, receiving information from the second of the two mobile programs, mobile programs service stations, extensions, or other applications concerning a transaction in which the second of the mobile programs, mobile program service stations, extensions, and other applications wishes to engage,

notifying the second mobile program, mobile program service station, extension, or other application of the one mobile program, mobile program service station, extension, or other application, and

enabling the two mobile programs, mobile program service stations, extensions, or other applications to communicate concerning the transaction.

14. (previously amended) The method of claim 13 in which the information is received from two mobile programs by a third mobile program.



15. (allowable) A method for enabling action by an operating environment in a distributed system with respect to a mobile program which is programmed in a language that is not fully supported by the operating environment, comprising

labeling a mobile program to identify operating environment features required for full support of the mobile program,

in an operating environment, examining the labeling of the mobile program to determine whether the operating environment supports all of the identified features, and

taking an action based on whether all the identified features are supported.

16. (allowable) The method of claim 15 wherein the action comprises sending the mobile program to another operating environment for processing.

17. (allowable) The method of claim 15 in which the action comprises retrieving non-program specific data from the mobile program.

18. (unchanged) A method for aiding communication with a mobile program executing in operating environments provided at nodes of a distributed system, comprising

maintaining a name space that uniquely identifies types of information to be interchanged as part of the communication, and

using a name within the name space to identify a type of information to be interchanged.

19. (unchanged) The method of claim 18 in which the mobile program registers an interface which includes the name of a type of information that is to be interchanged.

24. (previously amended) A method for controlling interaction between a mobile program and an application running in an operating environment provided at a node of a distributed system, comprising

defining a trusted portion of the operating environment which provides trusted services to the mobile program,

requiring portions of the application running in the operating environment to be registered as trusted, and

permitting indirect interaction via the operating environment between the mobile program and the application running in the operating environment only if the portions of the application required to be registered have been registered.

25. (previously amended) A method for enabling a mobile program to carry out defined functions including otherwise unsafe functions, through the use of extensions comprising

coding safe extensions to an operating environment and to an interpretive language under which the mobile program runs, and

permitting the mobile program to carry out the defined functions by making use of the extensions.

Add the following new claims:

--27. (new) The method of claim 9 in which the mechanism comprises a connector mechanism, and the objects comprise connector objects.--